First submitted: October 1998 (#846). Editor/reviewers please consult accompanying document with detailed responses to reviewer comments.

Interactive Virtual Relighting of Real Scenes

Céline Loscos, George Drettakis, Luc Robert

Abstract

Computer augmented reality (CAR) is a rapidly emerging field which enables users to mix real and virtual worlds. Our goal is to provide interactive tools to perform *common illumination*, i.e., light interactions between real and virtual objects, including shadows and relighting (real and virtual light source modification). In particular, we concentrate on virtually modifying real light source intensities and inserting virtual lights and objects into a real scene; such changes can be very useful for virtual lighting design and prototyping. To achieve this, we present a three-step method. We first reconstruct a simplified representation of real scene geometry using semi-automatic vision-based techniques. With the simplified geometry, and by adapting recent hierarchical radiosity algorithms, we construct an approximation of real scene light exchanges. We next perform a preprocessing step, based on the radiosity system, to create unoccluded illumination textures. These replace the original scene textures which contained real light effects such as shadows from real lights. This texture is then modulated by a ratio of the radiosity (which can be changed) over a display factor which corresponds to the radiosity for which occlusion has been ignored. Since our goal is to achieve a convincing relighting effect, rather than an accurate solution, we present a heuristic correction process which results in visually plausible renderings. Finally, we perform an interactive process to compute new illumination with modified real and virtual light intensities. Our results show that we are able to virtually relight real scenes interactively, including modifications and additions of virtual light sources and objects.

1 Introduction

Computer augmented reality (CAR) is a rapidly emerging field which enables users to mix real and virtual worlds. Many applications of augmented reality already exist, for example in entertainment, film and television or help-systems for repair and manufacturing. Using real scenes or images of real scenes has many advantages over traditional, manually modeled virtual environments: the great effort required to model detailed objects is avoided, the resulting complexity in geometric primitives is greatly reduced, and the user is confronted with familiar, real-world references, allowing immediate and intuitive immersion.

The focus of previous research in the domain has mainly been on registration and calibration [Azu97]. Our interest however focuses on *common illumination* between real and virtual objects, i.e. the interaction of light (shadows, reflections etc.) between real lights and objects and virtual lights and objects. Previous work in common illumination [SHC+96, NHIN86, YM98, JNP+95, Deb98, FGR93, DRB97] has provided solutions which handle certain cases of light interactions between real and virtual objects, and especially the case of virtual objects casting shadows onto real objects. Most of them focus on the production of high-quality, off-line generation of films, or are limited in the effects they produce.

In this paper, we present a system which allows *interactive* modification of real or virtual lights, based on a simple model of the real scene and recent developments in interactive hierarchical radiosity. Modification of real lights is a difficult problem because real shadows are already present in textures representing the real world, which are mapped onto the reconstructed models of real objects. Previous solutions [NHIN86, YM98] allow the virtual modification of the sun position in outdoors real environments, but are based on inherently non-interactive algorithms. We build our method on a previous interactive common illumination approach [DRB97], which was restricted in the effects it could treat to virtual objects casting shadows on reconstructed real surfaces. Concurrently with this work, two new solutions have been proposed [YDMH99, LFD+99]. We discuss how this work relates to these solution in Section 7.1.

Our solution has three main steps. We first reconstruct a 3D representation of a real scene, using advanced vision-based techniques. A hierarchical radiosity system is then initialized in a preprocess step, to represent real world illumination. The preprocess then automatically creates new *unoccluded illumination textures* which represent illumination without taking shadows into account, and finally reprojects real shadows using the modified radiosity estimation. The third step is an algorithm allowing interactive modification of real and virtual light source intensity.

For a lighting designer, this system provides realistic tools to experiment with the illumination of an enhanced real environment. All that is required is a few photographs of the real scene, the reconstruction of a small number of objects, and the system preprocess as will be described in this paper; the designer can then interactively manipulate real light intensities, or insert and manipulate virtual lights and objects. In Figure 1, an example of a modeled real scene is shown in (a). In (b), its real illumination was modified by switching off two lights. Moreover, a virtual light source was inserted into the real scene, modifying real object shadows. In (c), a virtual object was also inserted into the real scene, casting shadows onto real objects. This virtual object can be moved interactively at 3 frames per second (see accompanying web movie).

(a) (b) (c)

Figure 1: (a) Original real scene. (b) Virtual modification of the illumination of the real scene. (c) Real scene enhanced by a virtual object that moves at 3 frames per second.

In the following sections, we first present previous work in the several domains related to this work: augmented reality, 3D reconstruction of real scenes and hierarchical radiosity. Previous common illumination approaches are then discussed in more detail. We proceed to explain how we build a 3D geometric model representing the real scene, and present an overview of the

algorithm for interactive re-lighting. The preprocess phase is presented in detail, followed by a description of the interactive relighting process. We then describe results of relighting and interactive modification of real light intensities and the insertion of virtual lights and conclude with at discussion and future work.

2 Previous Work

Our work draws on multiple fields; in particular augmented reality, vision based reconstruction and global illumination. In the following, we will first give a rapid overview of augmented reality which concentrates, in general, on registration and calibration aspects. We next briefly discuss the 3D reconstruction method we use to build a simplified model of the real scene. We will use hierarchical radiosity to create a representation of real-world illumination, and also to permit interactive updates when moving virtual objects or modifying illumination; we thus introduce the basic concepts of this approach which are central to the understanding of our algorithm. We finally detail previous work on global common illumination, insisting in particular on the most closely related approaches which use radiosity methods.

2.1 Introduction to augmented reality

There are two main approaches to augmented reality: virtual and real environments can be combined by superimposing virtual objects on the real world viewed from semi-transparent glasses; alternatively, virtual and real environments can be merged with video images, and the result reprojected onto a screen. These two approaches are presented in detail in the survey of Azuma [Azu97] which also provides extensive references to related literature.

The first approach involves the calibration, registration and display of virtual objects in real time to avoid delays between projected images and the perceived real world. The second approach allows more interaction between real and virtual objects, because a geometric representation of the real scene is created from the images. We can therefore handle occlusion

between real and virtual objects, as well as visual effects such as *common illumination*, which is the interaction of light between virtual and synthetic objects. Nevertheless, achieving real time or interactive display of these effects remains a challenging problem.

2.2 Reconstruction of Real Models

The simulation of common illumination effects requires a geometric representation of the real world. Much research on the subject exists in the field of Computer Vision; we have chosen to use an advanced vision-based technique, which allows semi-automatic reconstruction based on multiple views.

The approach we use is presented in [FLR+97]. In order to build a representation of a real scene, several vision techniques are combined: automatic calibration of the camera, mosaicing, computation of the epipolar geometry which results in a polygonal reconstruction of the scene, and the projection of textures. The first step is the calibration of the camera which consists in retrieving the intrinsic parameters from a non-planar calibration pattern image using an automatic algorithm [Rob95]. The user provides approximate positions of 6 reference points. From this, the system retrieves intrinsic and extrinsic parameters of the camera. Then, four sets of three photographs each are taken, and a mosaic is built automatically for each set as presented in [ZFD97]. From the four mosaics, a 3D model is defined using the *TotalCalib* system [Tot] developed at the ROBOTVIS group. This system, shown in Figure 2, combines several techniques. Point correspondences are provided by a user, who clicks on one image to create a reference point. The matched points on the 3 other mosaics are given automatically by the system. From about 30 point correspondences, fundamental matrices are computed using a non-linear method [ZDFL95]. Polygonal regions are next manually selected by a user from point correspondences, and the system provides 3D coordinates of these polygons from the projection equations. Finally, textures are projected to allow correct perspective effects for a fixed viewpoint [FLR+97]. For each reconstructed polygon, a texture image is computed by

de-warping the original image (from a given viewpoint), and mapping it to the plane of the polygon.

Figure 2: The TotalCalib system to build 3D models of real scenes, using automatic calibration, and epipolar geometry.

The main advantage of such a system is that user intervention is restricted to the choice of reference matches and polygon vertex selection. This system is however not without limitations: the resulting model of the real scene is approximate and may contain artifacts, since there is no guarantee that geometric properties such as parallel edges or orthogonal angles will be preserved. This drawback can be removed by taking into account additional user input, as presented in the work of Debevec et al. [DTM96] or Poulin et al. [POF98b].

In the work by Debevec et al. [DTM96], reconstruction is based on a hierarchy of blocks. The main idea is to build polyhedra which include geometric constraints, such as parallelism, orthogonality, and size aspects. Polyhedra provide good approximations of many objects of the real world, especially for outdoor architectural scenes. This also allows the reconstruction of vertices which are invisible in the original images, but correspond to hidden vertices of the polyhedra. Another approach is described in [POF98b] in which the primitives are points, lines and polygons, and constraints such as parallelism, orthogonality, or co-planarity are determined

by the user.

2.3 Hierarchical radiosity

To achieve interactive relighting, we need an efficient description of light exchanges in the scene, including shadow information. We have chosen to use the hierarchical radiosity approach with clustering [HSA91, Sil95] with the extensions to dynamic environments [DS97]. We next introduce certain basic concepts of radiosity methods.

The radiosity method is based on energy exchanges, and has been used in computer graphics to simulate light interactions in synthetic environments [SP94], including indirect illumination. Since the radiosity method is a finite-element approach, a mesh representation of the scene is required, which is usually constructed with quadtrees.

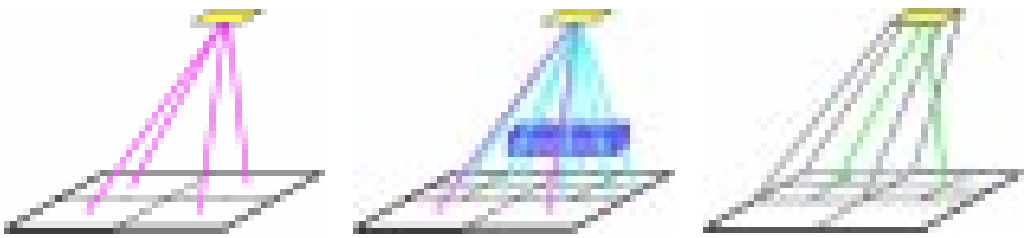
Hierarchical radiosity [HSA91] uses a multi-resolution representation of light, by creating a hierarchy of patches on each surface. Light exchanges are established at the appropriate levels at the patch hierarchy via a *link* data structure, resulting in an efficient solution. A generalization of this approach can be achieved using clusters [SAG94, SDS95], which represent groups of objects. The entire scene is contained in a single, "root" cluster. Clusters and patches can be linked at the appropriate level, depending on the refinement criterion, which decides whether the link represents the light transfer at a suitable, user defined, level of accuracy. If the light exchange is not sufficiently well-represented, the link is refined and the patches or clusters are then subdivided.

When links are established, the incoming irradiance is *gathered* at each patch, followed by a *push-pull* step performed to maintain a coherent multi-resolution representation of radiant exchanges [HSA91]. The cluster-based hierarchical radiosity starts with the root cluster linked to itself. The algorithm described by Sillion [Sil95] performs a refinement step, establishing links at appropriate levels followed by the gather and push-pull steps. Irradiance is pushed down to the leaves of the patch hierarchy, and radiosity is pulled up by averaging [Sil95]. This

is repeated until convergence.

Visibility information and form factors are stored with links. The visibility information can be of three types: *VISIBLE*, *INVISIBLE* or *PARTIAL*. When computing radiosity exchanges between two patches, the incoming irradiance is multiplied by the form factor and an *attenuation factor*, which varies from zero when the patches are mutually completely occluded, to one when the patches are entirely mutually visible. The attenuation factor represents the degree of occlusion between two patches. It is typically estimated by shooting rays between the two patches, and counting the percentage of rays blocked by occluders.

The hierarchical representation with links can be adapted to allow fast radiosity modification [DS97], by augmenting the links with a shaft data structure [HW91]. In addition, previously subdivided links, called *passive links* are maintained. The passive links contain all the necessary information allowing them to be reactivated at no cost, if it is required by a geometry change. See Figure 3 for an example.



(a) (b) (c)

Figure 3: (a) Original subdivision and links in purple. (b) Adding a dynamic object, and updating the hierarchy of elements and links. Eight links shown in blue were created. (c) The passive links with their shafts are maintained in the hierarchy, allowing fast identification of the dynamic object movement. In this case, two passive links shown in green were maintained. The corresponding shaft is outlined in grey.

2.4 Common illumination in augmented reality

The retrieval and simulation of common illumination between virtual and real objects has been treated by several researchers in previous work [SHC+96, NHIN86, YM98, JNP+95, Deb98,

FGR93, DRB97, YDMH99]. All use some form of a 3D representation of the real scene.

State et al. [SHC+96] use a composition of vision-based and magnetic tracking methods for accurate registration of the real environment. Virtual objects are inserted into a real scene and common illumination is performed, with a moving (real) point light source. Shadow maps are used allowing updates in real time, but only for direct illumination and sharp shadows from point sources.

Nakamae et al. [NHIN86] developed a solution for merging virtual objects into background photographs, and estimated the sun location to simulate common illumination effects in outdoors environments. More recently Yu [YM98] proposed a solution to virtually modify the illumination with different virtual positions of the sun in outdoors scenes. A pseudo-BRDF is first estimated, which is a function of the incident radiance on the reflected differential radiance.

Diffuse and specular reflectances are retrieved using multiple images from multiple viewpoints. From various virtual positions of the sun and from modified sky and environment illumination, modified outdoors illumination is performed pixel by pixel for each reconstructed triangle. However, for certain applications, an approximation of only the diffuse reflectance is sufficient.

For indoors environments, Jancène et al. [JNP+95] used vision-based techniques to retrieve the geometry of the real scene from a video sequence. Common illumination between virtual and real objects is simulated. This allows the creation of video sequences, with animated virtual objects such as a cloth, and the modification of the reflective properties of real objects. The final rendering is performed using a ray-tracing system, and images are merged using a masking algorithm.

Debevec [Deb98] also simulates common illumination effects using RADIANCE [War94], a ray tracing based global illumination system. In this work, the real environment is decomposed into the distant scene and the local scene. The distant scene is used to evaluate the global radiance, and the source emittance [DM97]. An approximate geometric model of the local scene is built using the methods previously developed by the same author [DTM96]. Since

radiance is accurately retrieved from images, rendering with mixed images is done by using the difference of the wanted effects and the original image value. This method can be adapted for indoors or outdoors environments.

The common illumination methods presented above are geared towards high-quality image generation, requiring in the order of minutes per frame. Those which allows relighting need several images under different lighting conditions, or several viewpoints.

Our approach is complementary: we want to use simple data, that is a single image of a single viewpoint under original lighting conditions, and from this we want to provide *interactive* common illumination effects, which will allow a designer to modify and experiment with different lighting conditions. Digital prototyping or "mock-ups" require this type of interactive capability; for a "final" high-quality animation, one of the previous methods can always be used.

Radiosity-based systems for common illumination

The most closely related previous work is that of Fournier et al. [FGR93] and its interactive extension [DRB97]. The system presented permits the retrieval of radiosity parameters from the textures extracted from the real scene images. In our approach, we use Fournier et al.'s basic derivations for the extraction of the quantities required to initialize the radiosity equations. We thus describe this work in more detail.

First the real scene is modeled manually, using a simplified representation. Given this model which is subdivided into patches, the reflectance can be extracted from the image textures. The reflectance of each patch i is chosen to be:

$$\rho_i = \frac{\hat{B}_i}{\hat{B}_A} \times \hat{\rho} \tag{1}$$

where $\hat{B}_i$ is the average intensity of the pixels in an image corresponding to the projected patch i, $\hat{B}_A$ the average intensity of the real image, and $\hat{\rho}$ is the average reflectance of the scene (given by the user). This estimation of the reflectance depends on the color of the texture (i.e., the

photograph of the real scene), which will be darker for patches in shadow. The emittance E_i of each source is estimated from the following equation:

$$\sum_i E_i A_i = (1 - \hat{\rho})\hat{B}_A \sum_i A_i \tag{2}$$

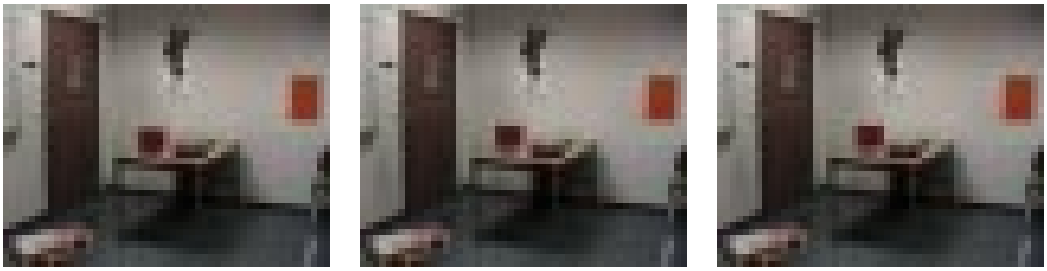
with A_i being the area of patch i. This approximation is based on the estimated ambient term in the progressive radiosity algorithm [CCWG88]. To simplify, and as it is approximately the case for our scenes, we consider that all the sources have the same intensity. However a system of equations could be solved for non-homogeneous intensities.

Once the reflectance(s) ρ_i and the emittance(s) E_i have been estimated, a progressive radiosity solution is applied. The result of this simulation is the radiosity B_i of each patch. The display is done using a *display correction factor* D_i of a patch i, which is first initialized to the radiosity B_i. When the scene is modified, the current radiosity B_i is updated to reflect the change. For example, if a virtual object is inserted, the patches on which a shadow is cast will have $B_i < D_i$. Modifications to the scene (notably the addition of virtual lights and objects), are performed by modulating the texture T_i of a pixel as follows:

$$\frac{B_i}{D_i} \times T_i \tag{3}$$

It is important to note here that the accuracy of the radiosity estimation B_i is irrelevant. Since a ratio is being used (which is 1 if there is no change), the only requirement is that the modifications to B_i have to be *consistent*. Note that ray-casting is used for rendering in [FGR93].

This approach was adapted in [DRB97], in the context of a hierarchical radiosity system, which allows common illumination between a dynamic virtual object and a real scene. The interactive update of the illumination when the virtual object moves uses the dynamic hierarchical radiosity solution described in [DS97]. An example of the results obtained by the method of [DRB97] is shown in Figure 4, where a red dynamic virtual object was inserted into a real scene, on the top of the desk. The shadows are virtually projected onto the table, using the display ratio described above (Eq. (3)).



(a) (b) (c)

Figure 4: (a) A virtual object, floating above the table, was inserted into a real scene using [DRB97], in 5.65 seconds. Shadows are projected onto the table using the display ratio of Eq. (3). (b) and (c) The dynamic object moves above the table. Links and radiosity are updated interactively in 3 frames per seconds.

2.5 Shortcomings of Previous Approaches

If we use the method of [FGR93, DRB97] to change the intensity of a real light source, the result is unsatisfactory. This can clearly be seen in Figure 5(b).

(a) (b)

Figure 5: (a) The original illumination of the real scene. Two sources (left and right) illuminate the wall, causing the shadow of the table to be cast on the wall. (b) Using the method of [DRB97], we turn off the left hand light; pre-existing shadows are not removed and the quality of the relighting is unsatisfactory: the shadow due to the left hand light is still clearly visible.

To see why, recall that the display is performed using the initial real world photograph [FGR93] or textures [DRB97]. The image or textures are then modulated by the ratio of the current radiosity value B_i (changed for example by turning off a light) over the originally com-

puted radiosity value D_i. Since the texture being modulated is a snapshot of the *real* global illumination in the scene, real shadows are already represented.

Consider the in Figure 5(b) for which the left hand light has been turned off. Observe the region in the blue square: it contains a region of the wall which was in shadow with respect to the left hand light, and a region which was not. Since the left hand light is turned off, the current radiosity value B_i will be reduced for both regions, by amounts which are very close in value since they vary only by the corresponding form factors. The textures of both regions are modulated by the ratio of this current radiosity B_i over the original radiosity value before the light was switched off. Since the texture (photo) corresponding to the region originally in shadow is much darker to begin with, the shadow will still be visible after the change. For the correct image to be displayed, we would need a way to make the texture in both shadowed and unshadowed have similar values.

This reveals the limitation of previous texture modulation approaches, which can only treat modifications to virtual objects or sources, since modification of real lighting conditions requires the modification of the original images or textures of the scene.

3 The Common Illumination System

The goal of our approach is to allow interactive modification of real source intensities, the insertion (and modification) of virtual sources, and the insertion and interactive manipulation of other virtual objects. All interactive updates will be performed with consistent update of shadows of real and virtual objects. Our system consists of 3 steps: 3D reconstruction of the real scene, a preprocessing initialization stage, and an interactive modification stage, during which the user can modify and enhance the real scene. The entire algorithm is summarized in Figure 6.

3D reconstruction
Build a simplified 3D model of the real scene
Preprocess
Hierarchical radiosity system set up
Refinement for shadow boundaries
Creation of the unoccluded illumination textures
System re–initialization and shadow reprojection
Additional preprocess for the insertion of virtual objects
Interactive modification
Modification of real and virtual lights.
Update when a virtual object moves.

Figure 6: Complete algorithm.

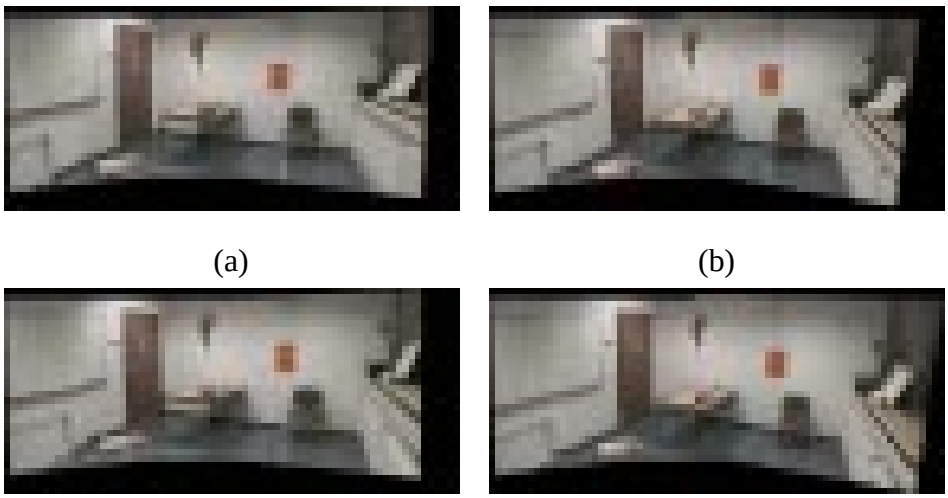
Representation of the real scene

The real scene is represented in our system with an approximation of its geometry and with projected textures. The model of the scene is built semi-automatically, using advanced vision techniques [FLR+97, DRB97] as described in Section 2.2. This process allows the reconstruction of the basic 3D model visible in the captured images (for example the mosaics shown in Figure 7).

The rest of the scene, which is not visible in the images, is built with points measured manually. Approximate textures are used to map the polygons of this part. The positions of the light sources are also measured manually and inserted into the 3D model.

The model is then an approximation of the captured room, with a more precise model for the visible part of the scene, and a coarse model for the rest. An example of the resulting reconstruction is shown in Figure 8.

A limitation of this approach is that the projection of the textures is done only for a single point of view. We are therefore restricted to viewing the scene from a static viewpoint. In Figure 8(a), the model is viewed from our system, and in (b) the complete model is shown, including the non-visible part.



(a) (b)

(c) (d)

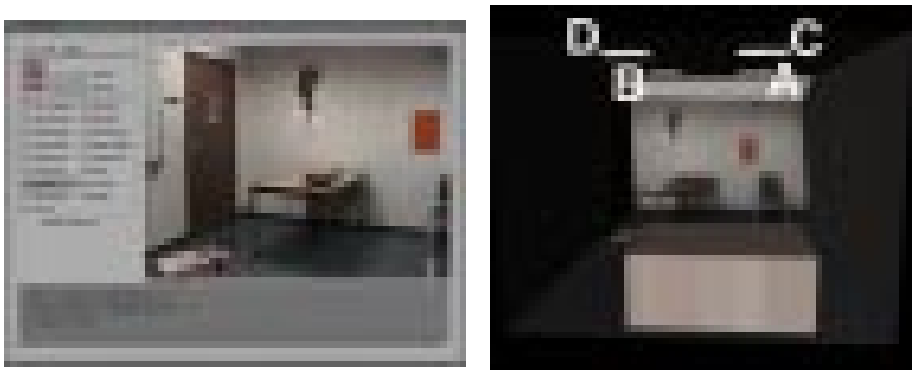
Figure 7: The four mosaics from four different points of view.

Preprocess to enable interactive re-lighting

The main contribution of our approach is the preprocessing algorithm which results in the generation of modified original textures, approximating *unoccluded* radiosity in the scene. We call these the *unoccluded illumination textures*. The original values of the textures, taken from the initial scene photograph are thus modified to represent illumination as if shadows of real objects where not taken into account. These shadows can be due to real light sources, or other secondary reflector objects.

Once we have created this unoccluded illumination texture, we can perform rapid relighting by modulating the texture with a ratio, corresponding to the increase or decrease in illumination due to lighting changes. This is achieved using a mesh of elements created by the radiosity algorithm. These elements are finer in the regions of shadow, and sufficient to capture other changes in illumination (for example due to indirect light).

The preprocess begins by setting up all necessary parameters for the estimation of the real scene illumination as in [DRB97]. A suitably subdivided mesh is essential for the appropriate modulation of texture; to achieve this a texture-based refinement is applied. To create the



(a) (b)

Figure 8: (a) The real scene viewed from our system. (b) The complete model including four lights (A, B, C, D).

unoccluded light textures, the information contained in the radiosity solution is used. Due to inaccuracies of the capture and reconstruction process, we use a heuristic correction to this process, based on shadow boundaries which are appropriately inferred from the radiosity solution. The result of this process are the unoccluded radiosity textures, which can then be modulated by the ratio of the final radiosity to unoccluded radiosity to reproject shadows and other illumination effects. This pre-process is explained in detail in the next section.

Virtual objects and virtual light sources can then be inserted if desired. The insertion of dynamic objects is performed using the method of [DRB97]. The algorithm used to insert virtual light sources is described in section 4.5.

Interactive Relighting

When the entire preprocessing step is completed, we can interactively modify the illumination of the light sources. The algorithm used is presented in section 5. Our interface, shown in Figure 9, allows the user to choose a new emittance in each RGB channel for real and virtual light sources.

A similar interface also exists for the insertion of real and virtual lights or objects (see web video for examples).

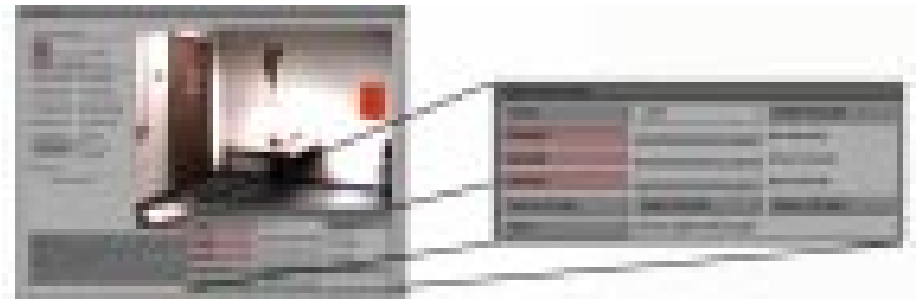


Figure 9: A screen snapshot of the interactive system. The user can manually select new light intensities for real or virtual light sources using the sliders shown in the inset.

4 Preprocessing for Virtual Interactive Relighting

Recall that our approach uses texture modulation as described in [FGR93, DRB97], and in Eq. (3) for rapid display, allowing the use of the graphics hardware.

As in [DRB97], we start by initializing the hierarchical radiosity system based on textures extracted from the original photographs, as presented in detail in Section 2.4.

To achieve the modification of real lighting we need to construct the unoccluded illumination textures, which are textures capturing an approximation of the illumination in the environment as if there were no occlusion from the light sources and secondary sources..

The creation of these textures has two steps: first we add in blocked light, using the information contained in the radiosity solution. Since this gives imperfect results due to the numerous approximations performed, a heuristic correction is applied by finding an appropriate reference patch which will give us a strong indication of the desired final color.

For both steps, it is important to have an appropriate mesh subdivision for radiosity, notably for the shadows on objects which are visible for our given viewpoint. We begin by describing our texture-based refinement, and then proceed to describe the two steps of the unoccluded illumination texture generation.

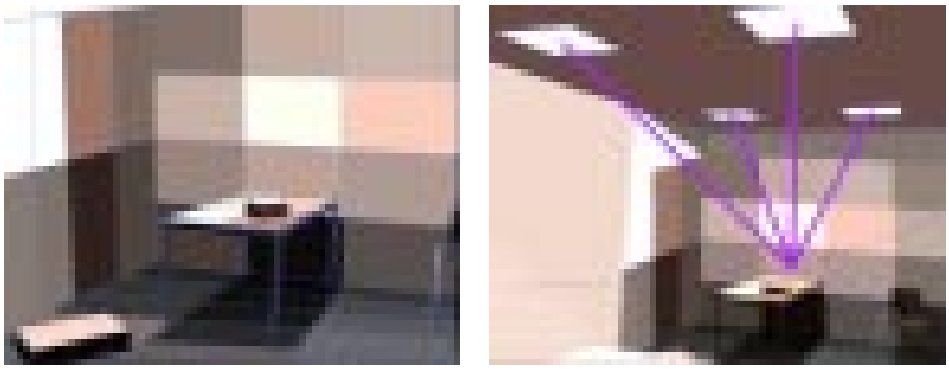
4.1 Texture-based refinement for shadow boundaries

If we use "standard" refinement criteria, such as BF refinement [HSA91] or error-driven refinement [GH96] we do not obtain suitable radiosity mesh subdivision. The main problem is that these approaches do not always guarantee good shadow boundaries (even when using the visibility factor of [HSA91]). In addition, the problem is more apparent in our case, since the geometry reconstruction and the visibility computation via ray-casting are not completely accurate. Discontinuity meshing [LTG93] is unsuitable for the same reasons, since discontinuity lines would be geometrically inaccurate. As a consequence, we use quadtree subdivision, with new, texture-based refinement.

The main idea is to use color information contained in original textures (i.e. the photos of the real scene reprojected as texture onto the reconstructed polygons), combined with the visibility information provided by the radiosity system as initialized above. Real shadows already exist in the textures, and correspond to regions which are darker. By using the visibility type (*VISIBLE, PARTIAL, OCCLUDED* see Section 2.3) contained in the links to patches in penumbra, and the color differences between neighboring patches, we can force refinement in regions corresponding to real shadow.

This refinement occurs after the first approximation of the radiosity solution of the real scene using the approach of [FGR93, DRB97]. The first radiosity solution is used to initialize several parameters such as reflectances and light source intensities. As shown in Figure 10(a), the initial subdivision, obtained using BF refinement, is coarse. Links have been attached to the leaves of the hierarchy of patches as in Figure 10(b), to provide accurately visibility information with respect to the source patches.

The texture-based refinement algorithm compares the visibility and the color of two neighboring leaves of the patch hierarchy. The visibility must be consistent with the color differences. We consider two cases, for a patch and each of its neighbors (the meaning of "similar" for color and visibility is defined below):



(a) (b)

Figure 10: (a) Coarse mesh before refinement. (b) All links are at leaves.

1. If the two patches have similar colors, they should also have the same visibility type with respect to all the real light sources. If it is not the case, then we subdivide the patch.

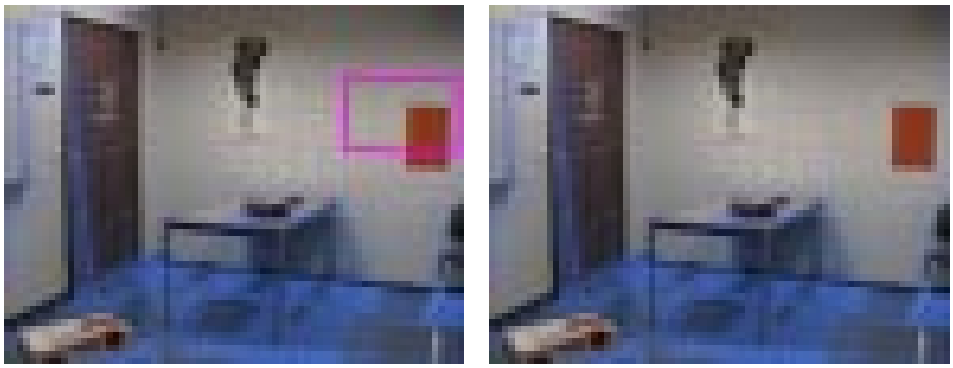
2. If the two patches have different colors, they should also have different visibility types. If not, we subdivide the patch.

If the patch has been subdivided, we examine the children created; if there is no change in visibility, the patch subdivision is cancelled and the patch is again a leaf of the patch hierarchy.

Case 1 occurs at the limits of shadow boundaries, and helps in producing finer elements in these regions. The process will stop when we reach the maximum subdivision level or when the patches are separated into visible and “in shadow”.

Case 2 occurs when ray-casting has failed to identify the correct visibility type. The patch may be unsubdivided however when the color difference is not due to a visibility change, but to a different texture. This is the case for the orange poster on the back wall in Figure 11(a).

Figure 12 shows how the refinement algorithm recursively traverses the hierarchy of elements and compares each pair of neighboring hierarchy leaves. We consider that the visibility is similar if the difference of the attenuation factor is less than a visibility threshold fixed by the user. Similarly, we consider two patches to have the same color if the distance in color is less than a color threshold also fixed by the user. To compute this distance, we first convert RGB



(a) (b)

Figure 11: (a) A patch in pink with a different color than neighbors but the same visibility. The patch was not subdivided. (b) Mesh after texture-based refinement with improved shadow boundaries. Compare to Figure 10.

values into LAB values. The distance between the two colors is simply computed by:

$$Distance_{Color}(i,n) = \sqrt{(L_i - L_n)^2 + (A_i - A_n)^2 + (B_i - B_n)^2} \quad (4)$$

Refinement for shadow boundaries
for each leaf *i*, compare with its neighbor leaves *n*
if *i* has a similar color to *n* **and** a different light source visibility
then subdivide *i*
else if *i* has a different color to *n* **and** similar light source visibility
then subdivide *i*
else do nothing
if *i* has been subdivided **then**
if *i* has no light source visibility differences with its children
then remove the subdivision of *i* (*i* is a leaf again)
else redo the process for each new child of *i*.

Figure 12: Texture-based refinement for shadow boundaries.

At the end of the refinement process, we set up all the necessary parameters: the reflectance, the display correction factor D^{orig}, which is equal to the original radiosity B^{orig}, and the texture T^{orig}, which is the original texture before any correction: i.e., extracted directly from the original photographs.

Links from real light sources are fixed at the leaves of the patch hierarchy. A radiosity step (gather/push-pull) is then computed, corresponding to this new subdivision.

The texture-based refinement results in well-defined shadow boundaries, which is very important for the subsequent texture modification step. The resulting refinement is shown in Figure 11(b).

4.2 Creating the Unoccluded Illumination Texture, Step 1: Adding in Blocked Light

Once an appropriate refinement has been performed for a radiosity solution, we can proceed with the creation of the unoccluded illumination textures. As mentioned above, the first step consists in "adding in" light blocked. The result of this step will be the generation of a modified texture, in which the blocked light has been incorporated.

We define $\bar{E}_{is}$ to be the irradiance from an source s blocked from patch i due to occlusion. A source is either a primary light source or a secondary source (i.e., a reflecting patch). This additional irradiance is the sum of the radiosity of each source times the form factor F_{is} and the complement of the attenuation factor equal to $(1 - V_{is})$ for each primary or secondary source s.

Considering each real source, we have the additional irradiance $\bar{E}_i$ for patch i:

$$\bar{E}_i = \sum_s F_{is}(1 - V_{is})E_{is} \tag{5}$$

The fact that all links are at the patch hierarchy leaves allows satisfactory estimation of $\bar{E}_i$, since the form-factor and visibility information are relatively accurate. For more accuracy, we take into account the occluded indirect illumination. However, since we have not reconstructed every object of the scene, and since the geometric model is approximate, the occluded irradiance due to indirect illumination is less accurate. In our tests the effect of adding in indirect light at this step has not been decisive.

To generate a new texture with the blocked light added, the original texture is modulated by a correction factor computed at the vertices of the leaf radiosity patches. Modulating the texture at patch vertices results in smooth modified textures.

The correction factor is based on the additional irradiance described above in Eq. (5). To include the blocked radiosity, we modulate the original texture T^{orig} as follows:

$$T_i^{inter} = \frac{\rho_i E_i + \rho_i \bar{E_i}}{D_i^{orig}} \times T^{orig} \tag{6}$$

In this equation, $\bar{E_i}$ is the potentially blocked irradiance (direct plus indirect), and $B_i^{orig} = D_i^{orig} = \rho_i E_i$. However, $\bar{E_i}$ is computed with the approximate values F_{is}, V_{is} and E_s, and thus the modulation of Eq. (6) is not sufficiently accurate.

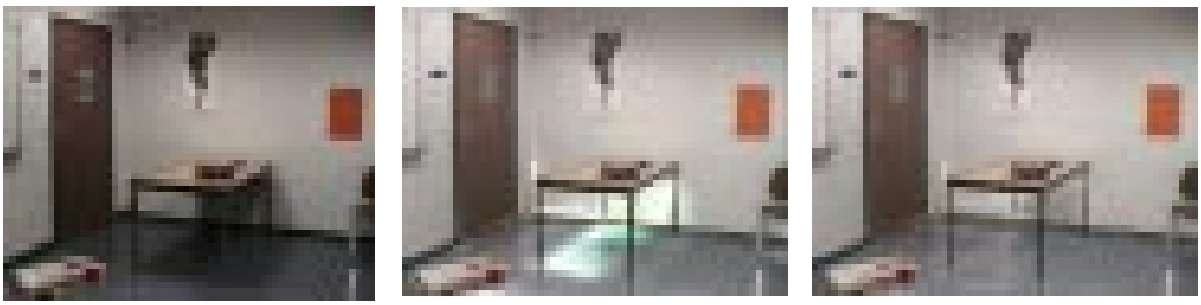
The intermediate texture T^{inter} is generated by rendering the leaves of the radiosity hierarchy with appropriate modulation values (Eq. (6)). If the modulation factor is greater than one, a multi-pass approach is used, as described in Appendix A.

In Figure 13 (b), we show an example of the texture generated after the addition of the blocked light, on the floor's original texture shown in (a). As can be clearly seen, the values computed are far too bright in the regions of shadow, due to the inaccuracies of the different processes used.

(a) (b) (c)

Figure 13: (a) Original texture. (b) The resulting texture T^{inter}, (c) The resulting texture T^{final}.

The texture obtained after this first step is used to update new reflectance values ρ_i^{inter}, extracted in the same manner as for the original photographs (Section 2.4). Radiosity values B_i^{inter} are updated using these new reflectance values, as well as the display correction factor



(a) (b) (c)

Figure 14: (a) Original texture. (b) The resulting texture T^{inter}, with real occluded illumination removed, mapped onto the geometry of the real scene. (c) The final texture T^{final} after the texture-based correction.

D_i^{inter} which is set equal to the newly computed radiosity plus the blocked light.

As was demonstrating in the example (Figure 14)(b), the resulting textures cannot be used "as is". To compensate for the inaccuracies of the initial step, a subsequent heuristic correction factor is applied, based on texture color.

4.3 Creating the Unoccluded Illumination Texture, Step 2: Texture color based correction

The intuition behind the heuristic correction step is to estimate the desired color of the unoccluded texture for a given element in shadow, based on a truly unoccluded element elsewhere on the same surface. We would like the color of each pixel of the occluded part of the texture to have a color similar to that of an unoccluded pixel. The similarity is modulated by the form factors since we want to keep unoccluded illumination effects in the final texture.

Consider a patch i in shadow, and a patch r chosen appropriately which is unoccluded. We want the corresponding texture values T_i and T_r to be equal, modulated by the form-factor, since the position of i and r with respect to the light sources is different. For the light sources S, the resulting desired value for the texture for i is as follows:

$$T_i = T_r \times \frac{\sum_s F_{is}}{\sum_s F_{rs}} \tag{7}$$

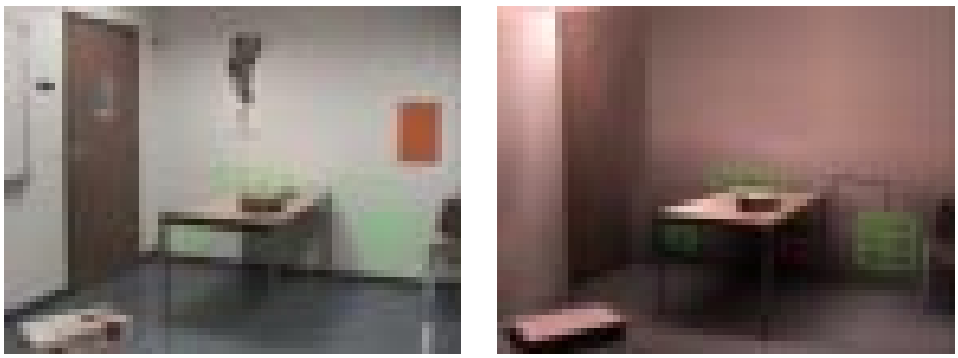
Since we operate in the context of polygon-based hardware rendering, we perform this correction on a per-patch basis. We modulate the texture of each patch using a correction factor. Instead of using the color of the texture, we use reflectance values which are stored with the radiosity system, and which are computed directly from textures. We associate to each occluded mesh radiosity element, a "reference patch" which will serve to correct the texture. For each radiosity mesh element in shadow, we thus attempt to find a corresponding unoccluded mesh element. We attempt to find a patch which is close, and which has similar reflectance values.

To do this, we first determine the frontier between occluded and unoccluded patches according to all light sources. Having all links at leaves ensures the classification *VISIBLE*, *INVISIBLE* or *PARTIAL* for a given patch with respect to a given source. We are therefore able to define a frontier composed of completely unoccluded patches that have occluded neighbors with respect to real light sources. This frontier usually encloses the regions where we need to modify the texture. However the algorithm does not depend on the creation of a closed region. The frontier elements will be used as references as explained below. From these selected elements, we keep only those which are visible from the viewpoint. This restriction is due to the view-dependent property of the textures we use. An example of such frontier patches is shown in Figure 15(a).

For each occluded patch i, we define a reference patch chosen in the frontier of unoccluded patches. The reference patch r is chosen to have a similar color as the occluded patch and to be at a minimum distance from i. For the occluded red patch, shown in Figure 15(b), the algorithm chooses the black patch as a "reference patch" from the frontier list of unoccluded elements shown in Figure 15(a). The black frontier patch is the closest patch that has a similar color to the red occluded patch (see the algorithm in Figure 16).

We define colors to be "similar" if the distance between them is less than a threshold defined by the user. As for the refinement, reflectances are converted into LAB values, and the distance $Distance_{Color}$ is computed using Eq. (4).

Once the reference patch has been chosen, we use Eq. (7), to determine the correction factor to be applied to the texture of the patch. Since the reference patch is at a certain distance from



(a) (b)

Figure 15: (a) Frontier in green composed of unoccluded patches, which enclosed shadow regions. (b) Black patch chosen in the frontier as a reference for the red selected patch in shadow.

the occluded patch, we modulate the reflectance of the reference patch by the ratio of the form factors F_{is} of patch i and F_{rs} of patch r with respect to the source s.

First, a corrected reflectance ρ_i^{corr} is computed:

$$\rho_i^{corr} = \rho_r \frac{\sum_s F_{is}}{\sum_s F_{rs}} \tag{8}$$

If no patch in the frontier of unoccluded elements is found for a certain patch i, then the reference patch is a default reference patch previously selected by the user on the polygon before the texture correction process.

Using the corrected reflectance, we generate the final unoccluded illumination texture. To generate this texture, we render the textured leaf patches of the patch hierarchy with an appropriate modulation factor, as when adding in blocker light.

For occluded patches only, the texture T_i^{inter} is modulated by the ratio of the correction factor ρ_i^{corr} of patch i over the intermediate reflectance ρ_i^{inter} computed directly from the intermediate textures T^{inter}:

$$T_i^{final} = \frac{\rho_i^{corr}}{\rho_i^{inter}} \times T_i^{inter} \tag{9}$$

If ρ_i^{corr} is greater than ρ_i^{inter}, we use a multi-pass display method described in the Appendix A, as for Step 1.

```
for each leaf i
   min_distance = ∞
   min_color = ε
   Reference = i
   for each patch in frontier list n
      if Distance(i, n) < min_distance
         and Distance_Color(n, i) < min_color
      then
         Reference = i
         min_distance = Distance(i, n)
         min_color = Distance_Color(i, n)
```

Figure 16: Algorithm to choose reference patches.

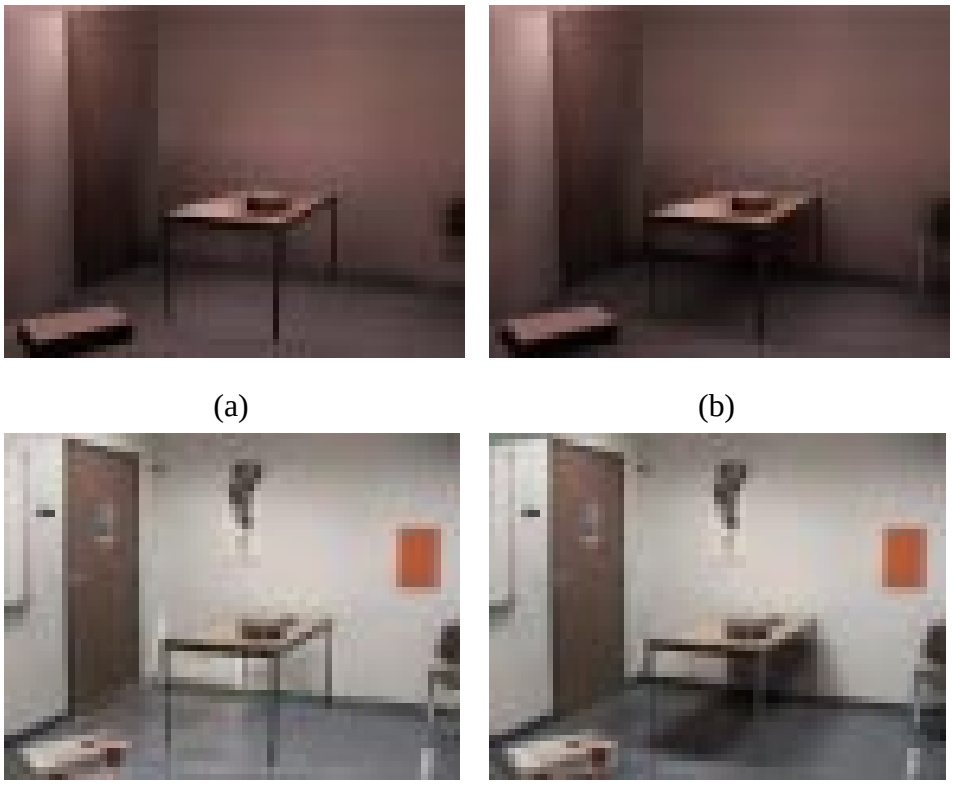
From this final unoccluded illumination texture T^{final}, we recompute new reflectance values ρ_i^{final} for occluded patches and perform a radiosity step, resulting in new radiosity values B_i^{final} based on the new reflectance. We then compute a new display correction factor D_i^{final}, equal to the new reflectance times the sum of the occluded irradiance E_i^{final} and the additional irradiance $\bar{E_i}$ (see Eq. (5)). Note that this display factor does not take into account shadow calculations.

An illustration of D_i^{final} is given in Figure 17(a), and B_i^{final} is shown in Figure 17(b). The result of the final textures is shown in Figure 17(c). Note that shadows have been mostly removed, and the texture does effectively represent illumination as if shadows had not been computed.

4.4 Shadow reprojection

After the steps previously described, we have a texture representing unoccluded illumination; we now need a way to (i) reproject original shadows and (ii) modify the intensity and add virtual objects and light sources.

This is achieved by modulating the unoccluded illumination texture T_i^{final} by the ratio $\frac{B_i^{final}}{D_i^{final}}$, which intuitively is the ratio of radiosity including shadow calculations over radiosity without shadows. Since B_i^{final} has a smaller value than D_i^{final} in regions of shadow, these areas are



(a) (b)

(c) (d)

Figure 17: (a) Display correction D^{final} corresponding to the new texture T^{final}. (b) Radiosity B^{final} corresponding to the new texture T^{final}. (c) The resulting final texture with shadows removed. (d) The resulting reprojection using these final values.

darker. As a result, shadows are appropriately reprojected, resulting in an image which is close to the original photograph. The result of this process is shown in Figure 17(d).

It is usually unnecessary to maintain the same subdivision used to modify the textures during preprocess, since it needs to very fine. For the interactive updates, this can be wasteful. Nonetheless, in some cases the mesh used for the preprocess is satisfactory.

To generate a coarser mesh, we clear everything previously computed, that is reflectances, radiosity, and the display correction factor. We also clear the subdivision and the link hierarchy. We then re-compute a solution based on a simple BFV refinement [HSA91], which results in a coarser mesh. To compute both B^{final} and D_i two radiosity solutions are actually performed.

At the end of the first solution, the display correction factor D_i is computed with all links at leaves of the hierarchy of mesh to allow accurate blocked light computation. A second radiosity solution is then computed, but keeping the same mesh; this permits the initialization of B^{final}, using less links. The resulting mesh is shown in Figure 18.

Figure 18: After the texture modification, a radiosity solution may be computed, using a BFV refinement. The resulting mesh is coarser in shadow regions than the one used to correct the texture.

4.5 Modified refinement to insert virtual sources

To treat the insertion of a virtual light source, we adapt the method of [DRB97], in which a virtual object can be inserted into the real scene and interactively manipulated [DS97]. This results in the projection of the shadows due to the virtual source on the real objects. The influence of a virtual light is often significant, and thus we force additional refinement by establishing all links to virtual sources on the polygons as opposed to allowing links from the virtual sources to the clusters. This is done on the polygons visible in the captured images; the polygons corresponding to the hidden parts of the scene are not affected by this forced refinement.

The additional light sources brighten the scene; again the multi-pass method of Appendix A is used to achieve this effect. Virtual light source insertion is illustrated in Figures 26, 21, 22, 24.

5 Final relighting

At this stage, we have completed the preprocess, and modifications are based on changes to the radiosity system. Links between patches and clusters in the radiosity hierarchy have already been established, including the form factor computation and the visibility determination. In order to achieve fast updates, the subdivision and the links are maintained during relighting. Since we only modify the intensity of the light sources, the subdivision and links still fit to the illumination even after modification. Keeping the same hierarchy may result in overly fine mesh subdivision if lights are switched off; since the user may switch them on again later however, we prefer to maintain the mesh subdivision. The modification process consists in recursively removing radiosity stored at each level of the hierarchy. We then perform a complete *radiosity step*: without performing additional refinement, we gather radiosity across the links, and perform the push-pull step to maintain a coherent representation of radiosity in the hierarchy. The iterative process is stopped when the global illumination is stable.

This process is interactive since the costly refinement step (which includes visibility and form-factor computation) is avoided. The update time depends on the initial level of subdivision. Note however that the insertion of a virtual object may result in additional subdivision. The update rate is the same if we modify one or several lights. Example update rates are shown in Figure 19, and discussed in the following section in detail.

6 Results

We have tested the algorithm for two different real scenes. For one of them, we use radiance images obtained using an adapted version of the algorithm of Debevec et al. [DM97] (see Appendix B). For each scene, we present results of relighting and adding virtual light and virtual objects, all performed interactively. All timings are reported on an SGI Onyx2 Infinite Reality workstation, R10000 running at 195Mhz.

The first scene is shown in Figure 20(a), under the original illumination. We first switch off

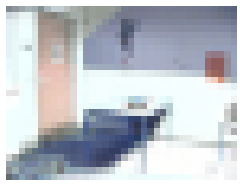
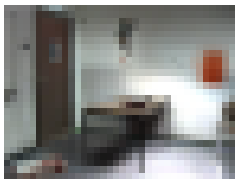
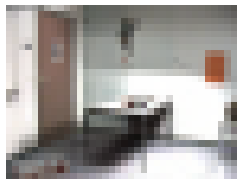
Refinement			
Decrease light intensity			
Times for modification	0.2 sec.	0.3 sec.	0.7 sec.
Times for display	0.2 sec.	0.2 sec.	0.6 sec.
Number of leaves/links	3486/11246	5781/16807	8247/50787

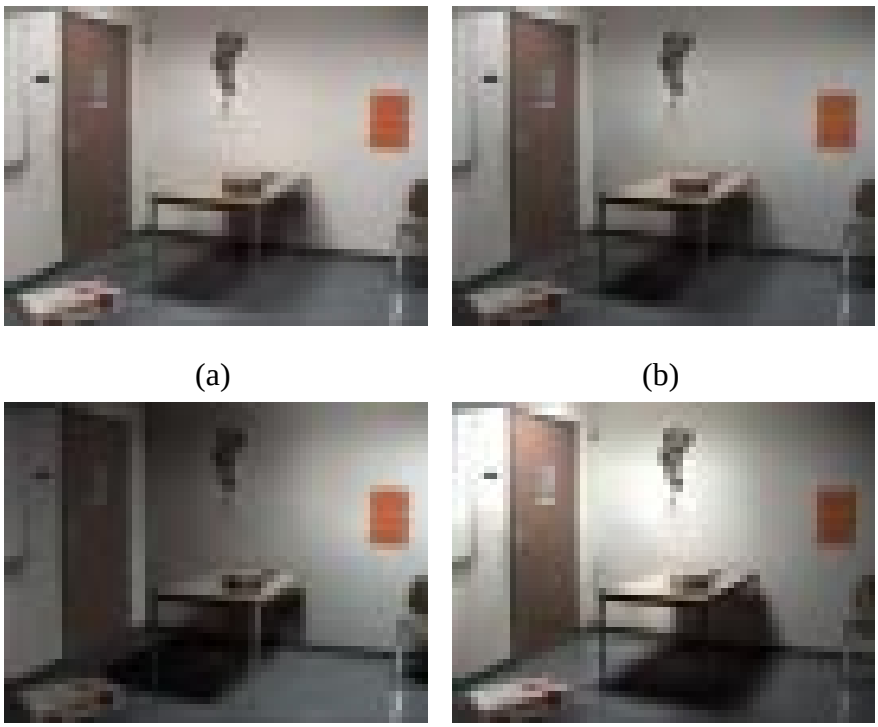
Figure 19: Interactive modification of a virtual light source intensity. The time rates depends on the level of the subdivision, and the number of established active links. The leaves at the bottom elements of the subdivision hierarchy.

the two back lights (C, D) shown in Figure 8. In the resulting image Figure 20(b), the scene is darker than the original illumination shown in Figure 20(a) but with no change in shadows.

We then switch off the front left light (A) and double the intensity of the right light (B) (see Figure 20(c)). The resulting shadow of the table is homogeneous in the umbra regions. As expected, the shadow due to the left light has disappeared, since the part of the scene which was illuminated by this light source is darker. Compare the new result with that of the method of [DRB97] previously shown in Figure 5(c), which was inexact, since real shadows were not removed from textures.

We now switch on the left light with double the original intensity and switch off the right light (see Figure 20(d)). Again, shadows are as expected (i.e. the shadow boundary of the right light is no longer visible). For each light modification, the whole process (radiosity step and

display) takes 0.8 seconds. The accompanying video[1] shows these light modifications, recorded in real time on an SGI Onyx2 Infinite Reality workstation.



(a) (b)

(c) (d)

Figure 20: (a) Original scene lit with shadow reprojection. (b) Back lights are switched off. (c) Left light is switched off, and right light has double intensity. (d) Right light is switched off, and left light has double intensity. Note that in this case, the mesh used for the texture correction was sufficient.

We can also insert a virtual source, and modify its intensity as described above. The insertion takes 7.8 seconds. An interesting test is to switch off all the real lights, and to illuminate the real scene only by a virtual source (see Figure 21(a) and (b)). Notice that real shadows from real light sources can no longer be seen. However, real objects such as the table cast new shadows on the floor and the walls, due only to the virtual light.

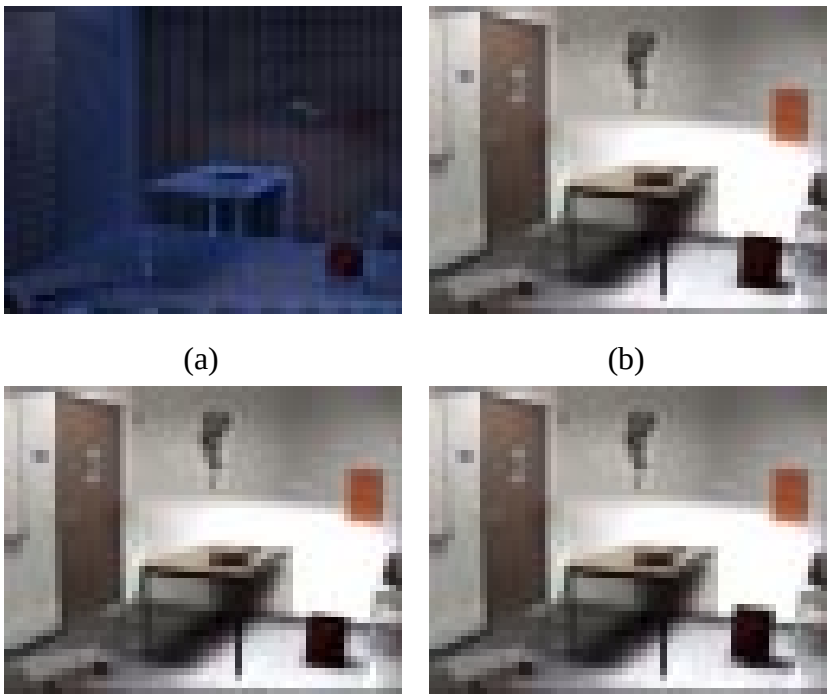
With this new illumination, we are still able to interactively move a dynamic virtual object, such as the orange box on the floor, previously inserted in 1.42 seconds, in Figure 22. Updates

[1] http://

(a) (b)

Figure 21: (a) Insert a virtual light. Switch off all real lights. The real scene is lit only by the virtual light. (b) Decrease the intensity of the virtual light.

take 0.3 sec. per frame, when moving the virtual object, with the subdivision shown in Figure 22(a). With both real and virtual illumination, this virtual object casts shadows onto the real scene.



(a) (b)

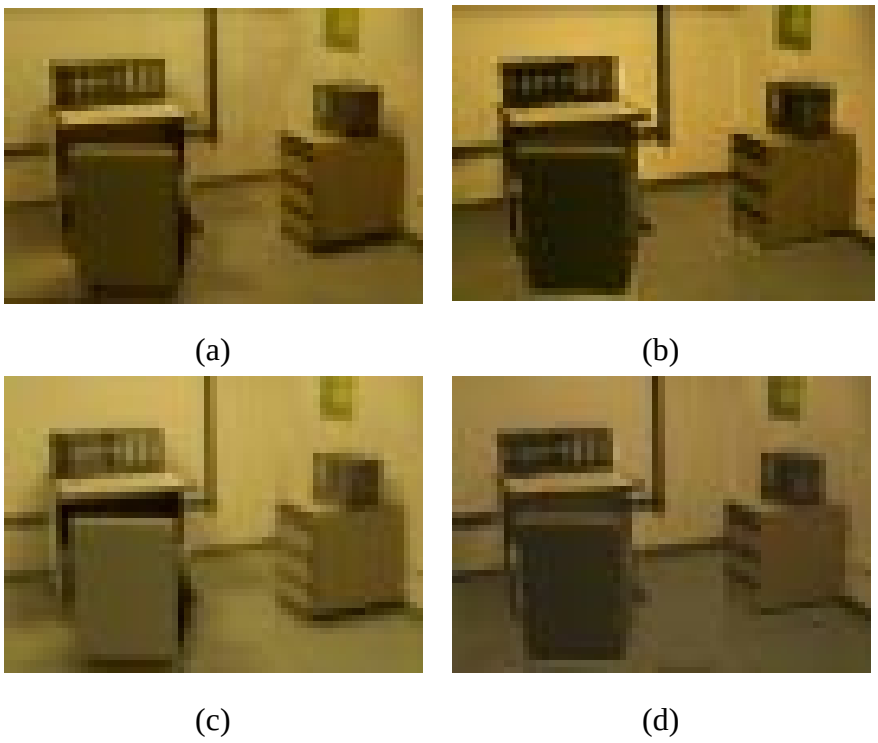
(c) (d)

Figure 22: (a) Insertion of virtual object and the consequent subdivision. (b), (c), (d) The red virtual object is moving at interactive rates.

We have also tested our method on another real scene[2], shown in Figure 23(a). In (b), we

[2]This real scene was modeled using the Rekon system developed at Montreal [POF98a]

have removed the real shadows from textures of this scene.



(a) (b)

(c) (d)

Figure 23: (a) Original real scene viewed from our system. (b) Real shadows were removed from real world textures. (c) Original real scene viewed from our system, using radiance images as textures. (d) Real shadows were removed from real world textures.

Another interesting test is to compare the results of our algorithm with real photographs in which we have turned off some of the real lights in the scene. In the right column ((a), (b), (c), (d)) of the Figure 24, we show the original photographs taken under different lighting conditions. On the left, we show the simulation resulting from our method for the same lighting conditions [3]. We have first performed real relighting in (e) by switching off the two back lights. In (f), we have switched off the right front light, and in (g), we have switched off the left front light. The reprojected shadows are softer than for the original picture. However, the overall lighting effect is similar. In (f), we inserted a virtual light, with all the original lights turned off. To test this scene, we took a photograph of the real scene using a real light which was used as a

[3]We have applied an appropriate scale-factor correction to account for the differences of overall lighting levels

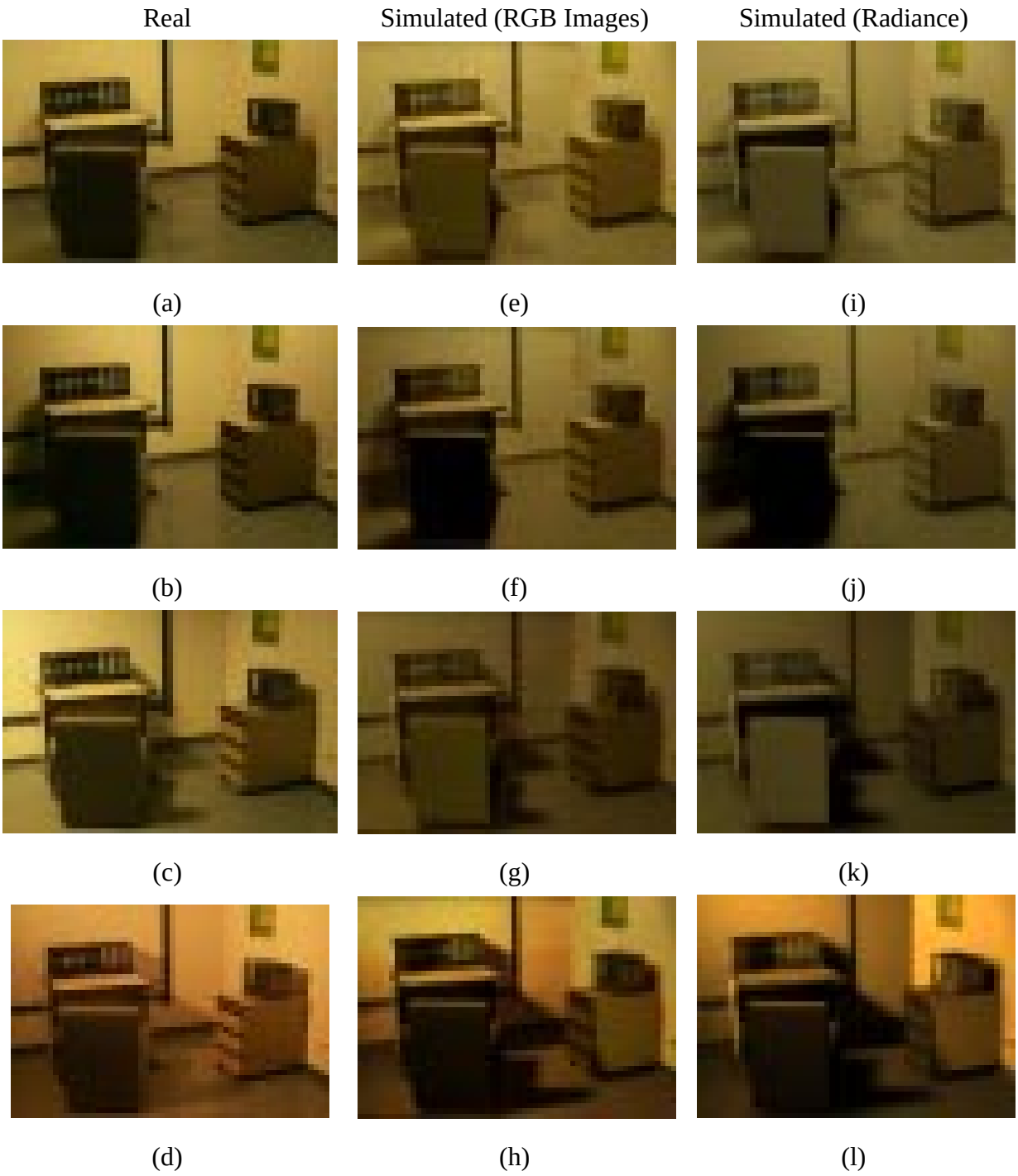


Figure 24: (a), (b), (c), (d) are real photograph taken under different lighting conditions. (e), (f), (g) and (h) are simulated images under respectively the same lighting conditions as the real photograph. (i), (j), (k) and (l) are the same simulation as above but using radiance images.

basis when modeling the virtual source. We show these images side by side.

For this scene, we performed the same modifications as above using radiance images as textures. The radiance images were obtained using the algorithm of Debevec et al. [DM97], adapted to our automatic camera[4], as described in the Appendix B. The results of texture modification are shown in Figure 23(c) and (d). Lighting modification has also been performed and the results, shown in Figure 24(i),(j),(k),(l), are very similar to those obtained using RGB images.

7 Discussion and Future work

Concurrently with this work, and since the original submission of this article, two new closely related methods have been developed. We briefly discuss how our approach relates to them, and then present our ideas for future work.

7.1 Discussion of more recent work

A recent method was developed by Yu et al. [YDMH99] which permits relighting for indoors environments, as well as the addition of virtual objects. In this method, reflectances are estimated accurately both for the diffuse and specular components, using a relatively large number of photographs (around 30-40) and user-controlled constrained lighting. However the rendering is computed using RADIANCE [War94], and is therefore far from interactive.

We [LFD+99] have also developed a completely different approach, which is based on taking several photographs of the real scene, using different user-controlled lighting conditions. This is used to estimate diffuse *reflectance*, for relighting and adding virtual objects. This method allows interactivity during scene modifications, but is based on ray-tracing and thus is subject to limitations in the number of sources and the resolution of the image.

[4]Kodak DC260

The approach we present here is in many ways complementary to the above. First, our approach has the simplest capture process since both other methods require user-controlled specific lighting, and more input photographs. In our approach we simply photograph the scene from a single viewpoint for the illumination processing[5]. Second, we do not attempt to perform a reflectance estimation, since we use a simple texture modulation approach for display. Finally, thanks to the use of the graphics hardware for display, we can achieve faster update rates, with fewer speed limitations.

It should be noted however that the approach of Yu et al. handles any viewpoint in the environment (albeit non-interactively), which is a significant advantage over our approach. The method in [LFD+99] notably allows the removal of real objects using texture generation on the estimated reflectance.

7.2 Future Work

Since the majority of the work is done during a preprocessing step, the relighting process is interactive and allows fast manipulation of the real scene. Two main issues need to be addressed: the speed of the updates and the quality of shadow removal.

The multi-pass display takes time, and it could also be optimized. The radiosity steps could also be optimized by avoiding the complete traversal of the hierarchy in the spirit of [DS97], when doing some relighting.

The quality of the shadow removal is directly related to the subdivision effected by the hierarchical radiosity algorithm. Our texture-based refinement has greatly improved the resulting quadtrees compared with traditional refinement approaches, but it is still prone to problems mainly due to inaccurate geometric reconstruction.

Another limitation of our system is the constraint to a fixed view point. Users modifying the illumination of a real scene, would like to change the point of view to better appreciate light effects. Building on recent work [POF98b, DBY98] we believe that we could develop a

[5] As with other methods of course, several photos are required for geometric reconstruction.

solution at least for a limited set of viewpoints.

Another interesting research direction is to allow the removal or replacement of real objects. The approach developed in [LFD+99] shows how to do this on a pixel-by-pixel basis. Even though the extension to our polygon-hardware based rendering is non-trivial, we believe that it is feasible.

8 Conclusion

In this paper we have addressed the difficult problem of common illumination for real and virtual scenes, and of virtual relighting. The problem is hard, since real-world photographs already contain real lighting effects such as shadows, which need to be correctly identified if they are to be modified.

We have presented a solution enabling interactive modification of both real and virtual illumination for reconstructed real scenes. The algorithm we presented has three main steps. The first step is the real scene reconstruction of a 3D geometric model using advanced vision-based techniques. The second step is a preprocessing algorithm. We first initialize a radiosity system, and use its structure to detect shadow regions, and appropriately refine the mesh. Once these regions are identified, we modify real world textures to include the real (primary and secondary) source illumination blocked by real occluders, thus brightening the dark shadow regions. After an additional heuristic correction, we can modulate these unoccluded illumination textures to reproject real shadows appropriately. The resulting simulated images are very similar to the original photographs.

We then have the ability to select and change the intensity of each real and virtual light source, and also add virtual objects into the scene. The results are convincing: shadow boundaries due to switched-off real lights disappear, and we can modify light source intensities, and move virtual objects. All modifications are interactive, and all diffuse common illumination effects such as shadows between real and virtual objects, are correctly represented.

Acknowledgments

This work was funded in part by the Reactive LTR Project ARCADE (#24944) of the European Union. We would like to thank Sylvain Bougnoux for providing the TotalCalib system. We also would like to thank Cyril Soler and François Sillion for the many discussions and help. Many thanks to Bruce Walter for his invaluable input on this project, and to the anonymous reviewers whose comments greatly improved the revised document and results.

References

[Azu97] Ronald T. Azuma. A survey of augmented reality. In *Presence: Teleoperators and Virtual Environments 6*, pages 355–385, August 1997. Earlier version in Course Notes #9: Developing Advanced Virtual Reality Applications, ACM SIGGRAPH (LA, 1995), 20-1 to 20-38.

[CCWG88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, August 1988.

[DBY98] Paul Debevec, George Borshukov, and Yizhou Yu. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques '98*, 9th EG workshop on Rendering, Vienna, Austria, June 1998. Springer Verlag.

[Deb98] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, 1998.

[DM97] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 369–378. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[DRB97] George Drettakis, Luc Robert, and Sylvain Bougnoux. Interactive common illumination for computer augmented reality. In J. Dorsey and P. Slusallek, editors, *Rendering Techniques '97*, pages 45–56, 8th EG workshop on Rendering, Saint Etienne, France, June 1997. Springer Verlag.

[DS97] George Drettakis and François Sillion. Interactive update of global illumination using A line-space hierarchy. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 57–64. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[FGR93] Alain Fournier, Atjeng S. Gunawan, and Chris Romanzin. Common illumination between real and computer generated scenes. In *Proceedings of Graphics Interface '93*, pages 254–262, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

[FLR+97] Olivier Faugeras, Stéphane Laveau, Luc Robert, Gabriella Csurka, Cyril Zeller, Cyrille Gauclin, and Imed Zoghlami. 3-d reconstruction of urban scenes from image sequences. *CVGIP: Image Understanding*, 1997.

[GH96] S. Gibson and R. J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, 1996. ISSN 0167-7055.

[HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.

[HW91] Eric Haines and John Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991.

[JNP+95] Pierre Jancène, Fabrice Neyret, Xavier Provot, Jean-Philippe Tarel, Jean-Marc Vézien, Christophe Meilhac, and Anne Vérroust. Res: computing the interactions between real and virtual objects in video sequences. In *Second IEEE Workshop on Networked Realities*, pages 27–40, Boston, Massachusetts (USA), October 1995. http://www-rocq.inria.fr/syntim/textes/nr95-eng.html.

[LFD+99] C. Loscos, M. C. Frasson, G. Drettakis, B. Walter, X. Grainer, and P. Poulin. Interactive virtual relighting and remodeling of real scenes. In *Rendering Techniques '99*, New York, NY, June 1999. Springer Wien. To appear.

[LTG93] Daniel Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 199–208, 1993.

[NHIN86] Eihachiro Nakamae, Koichi Harada, Takao Ishizaki, and Tomoyuki Nishita. A montage method: The overlaying of the computer generated images onto a background photograph. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 207–214, August 1986.

[POF98a] P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Rendering Techniques '98 (9th Eurographics Workshop on Rendering)*, pages 93–104. Springer-Verlag, June 1998.

[POF98b] Pierre Poulin, Mathieu Ouimet, and Marie-Claude Frasson. Interactively modeling with photogrammetry. In *Rendering Techniques '98*, 9th EG workshop on Rendering, Vienna, Austria, June 1998. Springer Verlag.

[Rob95] L Robert. Camera calibration without feature extraction. *Computer Vision, Graphics, and Image Processing*, 63(2):314–325, March 1995. also INRIA Technical Report 2204.

[SAG94] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[SDS95] F. Sillion, G. Drettakis, and C. Soler. A clustering algorithm for radiance calculation in general environments. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.

[SHC+96] Andrei State, Gentaro Hirota, David T. Chen, Bill Garrett, and Mark Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 429–438. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[Sil95] F. X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):240–254, September 1995. ISSN 1077-2626.

[SP94] François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994. excellent coverage of radiosity and global illumination algorithms.

[SS98] Cyril Soler and Francois Sillion. Automatic calculation of soft shadow textures for fast, high-quality radiosity. In *Rendering Techniques '98*, 9th EG workshop on Rendering, Vienna, Austria, June 1998. Springer Verlag.

[Tot] http://www.inria.fr/robotvis/personnel/sbougnou/TotalCalib/.

[War94] Gregory J. Ward. The RADIANCE lighting simulation and rendering system. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 459–472. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[YDMH99] Y. Yu, P.E. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99 (to appear)*, 1999.

[YM98] Yizhou Yu and Jitendra Malik. Recovering photometric properties of architectural scenes from photographs. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, 1998.

[ZDFL95] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78:87–119, October 1995.

[ZFD97] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Puerto Rico, June 1997. IEEE.

A Multipass Display

In our algorithm, the display is perform by setting a ratio B_i/D_i as a color, and modulate the texture by it, as described in Eq. (3). This display is insufficient when B_i is greater than D_i. This is due to a limitation of the *glColor* function of OpenGL, which requires a color value between zero and one. Therefore, if no special treatment is done when B_i/D_i is greater than one, the ratio is automatically clipped to one, and the desired illumination effect is not achieved.

```
Multi-pass display
  glBegin(GL_BLEND)
  glTexEnvf(GL_TEXTURE_ENV,
       GL_TEXTURE_ENV_MODE,
       GL_MODULATE)
  NumIter = 1
  MaxIter = DrawScene(NumIter)
  glBlendFunc(GL_ONE, GL_ONE)
  while NumIter < MaxIter do
    DrawScene(NumIter)
    glBlendFunc(GL_ONE, GL_ONE)
    NumIter++
  glDisable(GL_BLEND)
```

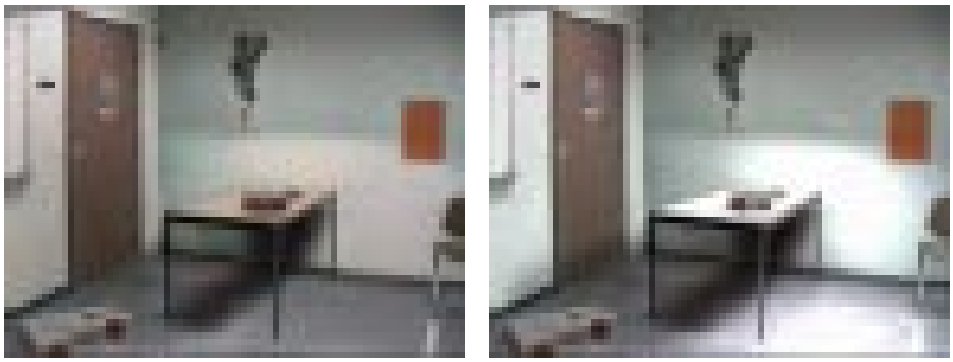
```
DrawScene ( NumIter )
  MaxIter = 0
  for each leaf
    if(NumIter == 0) then
      Maxiter = Max (Maxiter, Bi / Di)
    glTexImage2D( ..., Texture)
    for each vertex
      color = (Bi / Di) - NumIter
    if color > 1 then color = 1
    if color < 0 then color = 0
    glColor(color)
    return Maxiter
```

Figure 25: Multi-pass display.

In order to brighten the textures when source intensities increase, we use a n-pass display method, where $(n-1)$ is the integer part of B_i/D_i. The n^{th} pass corresponds to the remainder of $B_i/D_i-(n-1)$. To achieve the desired effect, we use the *glBlend* function of OpenGL. This enables us to modulate the texture by the color B_i/D_i-i or 1 if $(B_i/D_i-i)>1$, n times. The multi-pass algorithm is described in Figure 25. The drawback of this approach is that the display time is increased during the light modification pass, taking between a 0.05 and 1.0 seconds.

An example of the multi-pass display is shown in Figure 26, where a virtual light source has been added to the scene.

Similar problems have been encountered by other researchers. Debevec et al. [DBY98] use a similar approach when combining several weighted textures to create a single one, while Soler and Sillion [SS98] also used a multi-pass display to correctly modulate textures with direct illumination.



(a) (b)

Figure 26: (a) Insertion of virtual light source, without multi-pass display. (b) Using the multi-pass display (5 passes in 0.3 seconds).

B Radiance Images

The use of RGB textures extracted from a camera has several problems, including the limited dynamic range and loss of information due to saturation. Instead of working with RGB textures, we tested our algorithm using radiance textures provided by high–dynamic range photographs [DM97].

Our goal was to adapt this approach for use with a reasonably priced digital camera such as the Kodak DC260 which costs ten times less than professional level digital cameras. The problem with cameras in this price range is the lack of precise control of shutter speed. To create these radiances images, we thus had to adapt the algorithm of Debevec et al. [DM97]

Despite the lack of explicit shutter speed control, our camera provides nine different picture settings using the ”EV” parameter. When EV equals zero, the camera automatically chooses the parameters (aperture and shutter speed) that fit the lighting. When EV is negative, the camera chooses faster shutter speeds, and when it is positive, it chooses slower shutter speeds.

To apply the Debevec et al. algorithm, we convert each possible EV values into shutter speeds according to a reference time t, set for $EV = 0$. We based our assumptions on traditional F-stop ranges(2^{EV}).

EV	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
exposure time	t/4	t/2$\sqrt{2}$	t/2	t/$\sqrt{2}$	t	$\sqrt{2}$t	2t	2$\sqrt{2}$t	4t

The results of this adapted algorithm are quite satisfactory. The camera response function extracted seems reasonable, and we avoid the problems of saturation and low dynamic range of traditional RGB images.